Funciones: guía de ejercicios

1. Escribir una función que imprima el mensaje "Hello World!"

```
>> Hello world!
```

2. Escribir una función que no reciba parámetros, que en su cuerpo genere dos variables `mi_numero` y `mi_numero_2` con valores `10` y `20` y que imprima la suma. El resultado debería ser:

```
>> La suma es: 30
```

3. Escribir una función que reciba un parámetro (o argumento). La función tiene que imprimir un mensaje reportando cuál es el valor del parámetro que recibió. Por ejemplo, si recibe un `string` "esto es un texto", la función debe imprimir

```
>> "he recibido como parametro: esto es un texto"
```

4. Dada la siguiente función:

```
def mi_funcion(a, b):
    x = a + b
    return x
```

```
i. Escribir un código que imprima la suma de dos números
ii. Escribir un código que imprima la concatenación de dos strings
```

5. Escribir una función que reciba un número como argumento y que imprima un mensaje donde se muestre el numero ingresado y seexplique si el número es par o impar. Por ejemplo, si la función se llama `mi_funcion` entonces:

```
mi_funcion(5)
>> "El numero 5 es impar"

mi_funcion(4)
>> "El numero 4 es par"
```

6. Tomar la función definida en el punto anterior y agregar el siguiente comportamiento: si el argumento pasado a la misma no es un entero `int` entonces tiene que imprimir un mensaje que diga "Argumento incorrecto" y explique el tipo de argumento recibido, por ejemplo:

```
mi_funcion([1,23,4])
>> "Argumento incorrecto: <class 'list'>"

mi_funcion("4")
>> "Argumento incorrecto: <class 'str'>"
```

7. Implementar una función que reciba dos argumentos. Si por lo menos uno de los dos argumentos no son strings `str` entonces que devuelva el valor `False`, en cambio si los dos argumentos son strings, que devuelva el conjunto de caracteres que consituyen ambos argumentos, en mayúsculas. Por ejemplo, si la función se llama `mi_funcion`:

```
res = mi_funcion(1, "bowie")
print(f"el resultado es: {res}")
>> el resultado es: False

res = mi_funcion(1, [])
print(f"el resultado es: {res}")
>> el resultado es: False

res = mi_funcion("a", "b")
print(f"el resultado es: {res}")
>> el resultado es: {'A', 'B'}

res = mi_funcion("david", "bowie")
print(f"el resultado es: {res}")
>> el resultado es: {'V', 'B', 'O', 'D', 'E', 'A', 'I', 'W'}
```

8. Implementar 5 casos de test para una función que: dado un texto (`str`), si el mismo tiene más de 10 caracteres , tiene que devolver el mismo texto pero en mayúsculas. En otro caso, si tiene un número par de caracteres tiene que devolver un conjunto con todos los caracteres utilizados y si tiene un numero impar de caracteres tiene que devolver el número de caracteres. Por ejemplo, si la función se llama `mi_funcion` , dos tests podrían ser:

```
def mi_funcion(una_palabra):
    return None

# Test 1
resultado = mi_funcion("hoy es un dia soleado en Buenos Aires")
if resultado == "HOY ES UN DIA SOLEADO EN BUENOS AIRES":
    print("test correcto")
else:
    print("test incorrecto")

# Test 2
resultado = mi_funcion("a1b2")
if resultado == {"a", "1", "b", "2"}:
    print("test correcto")
else:
    print("test incorrecto")
```

9. Implementar una función que pase todos los casos de tests escritos en el punto anterior.